SQL PROJECT

PIZZA SALES

HELLO !

My name is Ritlesh Kumar and in this project, I have utilized SQL queries to solve business questions that were related to pizza sales.

QUESTION

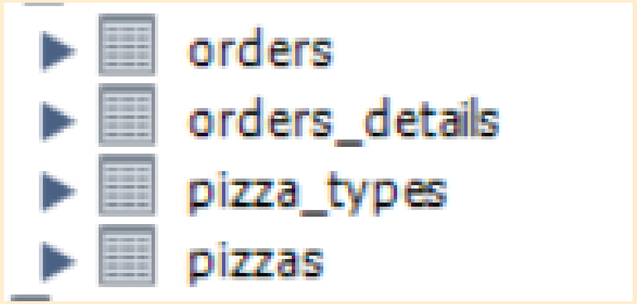
- *RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.*
- *CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.*
- *IDENTIFY THE HIGHEST-PRICED PIZZA.*
- *IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED.*
- *LIST THE TOP 5 MOST ORDERED PIZZA TYPES ALONG WITH THEIR QUANTITIES.*

- *JOIN THE NECESSARY TABLES TO FIND THE TOTAL QUANTITY OF EACH PIZZA CATEGORY ORDERED.*
- *DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY.*
- *JOIN RELEVANT TABLES TO FIND THE CATEGORY-WISE DISTRIBUTION OF PIZZAS.*
- *GROUP THE ORDERS BY DATE AND CALCULATE THE AVERAGE NUMBER OF PIZZAS ORDERED PER DAY.*
- *DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE.*

- *CALCULATE THE PERCENTAGE CONTRIBUTION OF EACH PIZZA TYPE TO TOTAL REVENUE.*
- *ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME.*
- *DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE FOR EACH PIZZA CATEGORY.*

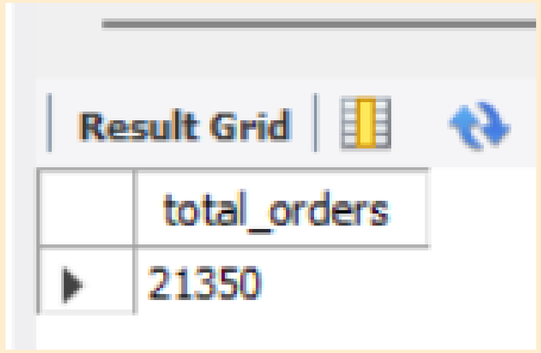
DATA

This project uses a relational Pizza Sales database consisting of four tables: Orders, Order Details, Pizzas, and Pizza Types.
The database captures order timing, pizza details, pricing, and categories, enabling detailed sales and revenue analysis using SQL.
It is designed to efficiently analyze customer behavior, popular pizzas, and business performance.



RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

```
SELECT
    COUNT(order_id) AS total_orders
FROM
    orders;
```



CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.

```sql
SELECT
    ROUND(SUM(orders_details.quantity * pizzas.price),
            2) AS total_revenue
FROM
    orders_details
        JOIN
    pizzas ON pizzas.pizza_id = orders_details.pizza_id
```

Result Grid

	total_revenue
▶	817860.05

MySQL

CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.

```sql
SELECT
    ROUND(SUM(orders_details.quantity * pizzas.price),
            2) AS total_revenue
FROM
    orders_details
        JOIN
    pizzas ON pizzas.pizza_id = orders_details.pizza_id
```

Result Grid

	total_revenue
▶	817860.05

MySQL

IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED

```sql
SELECT
    pizzas.size,
    COUNT(orders_details.order_detail_id) AS order_count
FROM
    pizzas
        JOIN
    orders_details ON pizzas.pizza_id = orders_details.pizza_id
GROUP BY pizzas.size
ORDER BY order_count DESC;
```

Result Grid | Filter

size	order_count
L	18526
M	15385
S	14137
XL	544
XXL	28

LIST THE TOP 5 MOST ORDERED PIZZA TYPES ALONG WITH THEIR QUANTITIE

```sql
SELECT
    pizza_types.name, SUM(orders_details.quantity) AS quantity
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    orders_details ON orders_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY quantity DESC
LIMIT 5;
```

Result Grid | Filter Rows:

name	quantity
The Classic Deluxe Pizza	2453
The Barbecue Chicken Pizza	2432
The Hawaiian Pizza	2422
The Pepperoni Pizza	2418
The Thai Chicken Pizza	2371

MySQL

JOIN THE NECESSARY TABLES TO FIND THE TOTAL QUANTITY OF EACH PIZZA CATEGORY ORDERED.

```
SELECT
    pizza_types.category,
    SUM(orders_details.quantity) AS quantity
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    orders_details ON orders_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.category
ORDER BY quantity DESC;
```

Result Grid | Filter Rov

category	quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

MySQL

DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY

```sql
SELECT
    HOUR(order_time) AS hour, COUNT(order_id) AS order_count
FROM
    orders
GROUP BY HOUR(order_time);
```

Result Grid

hour	order_count
11	1231
12	2520
13	2455
14	1472
15	1468
16	1920
17	2336
18	3200

GROUP THE ORDERS BY DATE AND CALCULATE THE AVERAGE NUMBER OF PIZZAS ORDERED PER DAY

```sql
SELECT
    ROUND(AVG(quantity), 0) as avg_pizza_ordered_per_day
FROM
    (SELECT
        orders.order_date, SUM(orders_details.quantity) AS quantity
    FROM
        orders
    JOIN orders_details ON orders.order_id = orders_details.order_id
    GROUP BY orders.order_date) AS order_quantity;
```

Result Grid | Filter Rows:

	avg_pizza_ordered_per_day
▶	138

JOIN RELEVANT TABLES TO FIND THE CATEGORY-WISE DISTRIBUTION OF PIZZAS.

```
select category , count(name) from pizza_types
group by category;
```

Result Grid | Filter R

	category	count(name)
▶	Chicken	6
	Classic	8
	Supreme	9
	Veggie	9

MySQL

CALCULATE THE PERCENTAGE CONTRIBUTION OF EACH PIZZA TYPE TO TOTAL REVENUE.

```sql
SELECT
    pizza_types.category,
    ROUND(SUM(orders_details.quantity * pizzas.price) / (SELECT
                    ROUND(SUM(orders_details.quantity * pizzas.price),
                                2) AS total_revenue
                FROM
                    orders_details
                        JOIN
                    pizzas ON pizzas.pizza_id = orders_details.pizza_id) * 100,
            2) AS revenue
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    orders_details ON orders_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.category
ORDER BY revenue DESC;
```

Result Grid | Filter R

category	revenue
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE.

```sql
SELECT
    pizza_types.name,
    SUM(orders_details.quantity * pizzas.price) AS revenue
FROM
    pizza_types
        JOIN
    pizzas ON pizzas.pizza_type_id = pizza_types.pizza_type_id
        JOIN
    orders_details ON orders_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY revenue DESC
LIMIT 3;
```

Result Grid | Filter Rows:

name	revenue
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5

ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME.

```sql
select order_date,
sum(revenue) over(order by order_date) as cum_revenue
from
(select orders.order_date,
sum(orders_details.quantity * pizzas.price) as revenue
from orders_details join pizzas
on orders_details.pizza_id = pizzas.pizza_id
join orders
on orders.order_id = orders_details.order_id
group by orders.order_date) as sales;
```

Result Grid | Filter Rows:

order_date	cum_revenue
2015-01-01	2713.8500000000004
2015-01-02	5445.75
2015-01-03	8108.15
2015-01-04	9863.6
2015-01-05	11929.55
2015-01-06	14358.5
2015-01-07	16560.7
2015-01-08	19399.05

MySQL

DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE FOR EACH PIZZA CATEGORY.

```sql
select name, revenue from
(select category, name, revenue,
rank() over(partition by category order by revenue desc) as rn
from
(select pizza_types.category, pizza_types.name,
sum(orders_details.quantity * pizzas.price) as revenue
from pizza_types join pizzas
on pizza_types.pizza_type_id = pizzas.pizza_type_id
join orders_details
on orders_details.pizza_id = pizzas.pizza_id
group by pizza_types.category, pizza_types.name) as a) as b
where rn <= 3;
```

Result Grid | Filter Rows:

name	revenue
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5
The Classic Deluxe Pizza	38180.5
The Hawaiian Pizza	32273.25
The Pepperoni Pizza	30161.75
The Spicy Italian Pizza	34831.25
The Italian Supreme Pizza	33476.75

MySQL

CONTACT INFORMATION

- Phone : 6201493881
- Gmail : ritleshkumar03@gmail.com
- LinkedIn : Ritlesh Kumar


THANK YOU